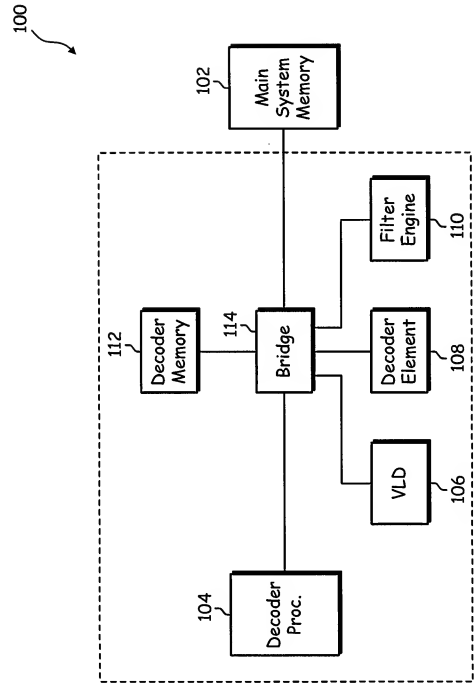
100
102
Main System Memory
104
Decoder Proc.
106
VLD
108
Decoder Element
110
Filter Engine
112
Decoder Memory
114
Bridge

FIG. 1

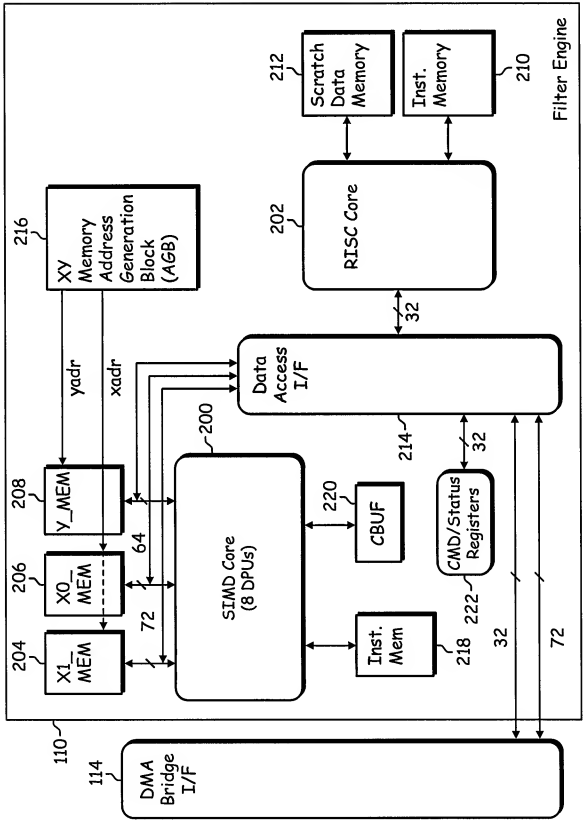
Filter Engine
110
DMA Bridge I/F
114
X1_MEM
204
XO_MEM
206
Y_MEM
208
72
64
SIMD Core (8 DPUs)
200
Inst. Mem
218
CBUF
220
CMD/Status Registers
222
32
72
Data Access I/F
214
32
RISC Core
202
Scratch Data Memory
212
Inst. Memory
210
XY Memory Address Generation Block (AGB)
216
yadr
xadr

FIG. 2

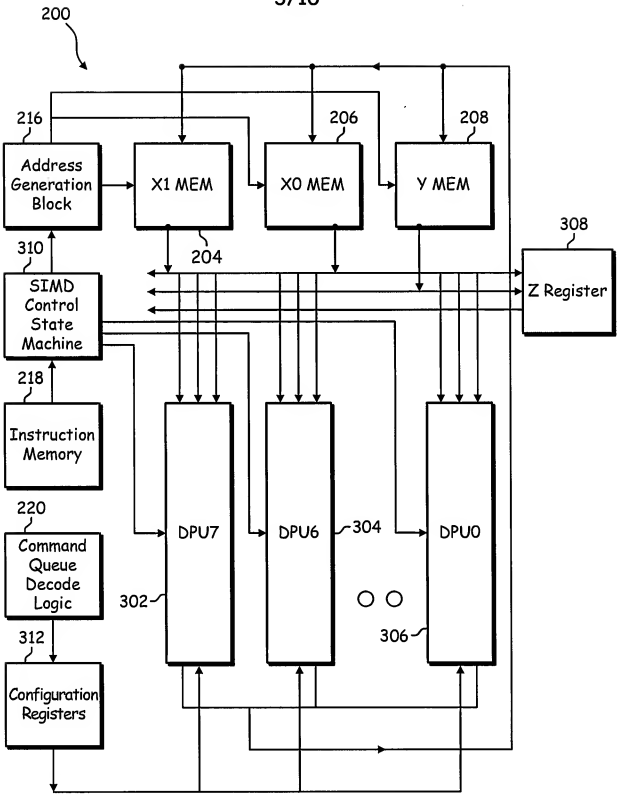
200
216
Address Generation Block
X1 MEM
206
X0 MEM
208
Y MEM
308
Z Register
310
SIMD Control State Machine
204
218
Instruction Memory
220
Command Queue Decode Logic
312
Configuration Registers
DPU7
DPU6
304
DPU0
302
306

FIG.3

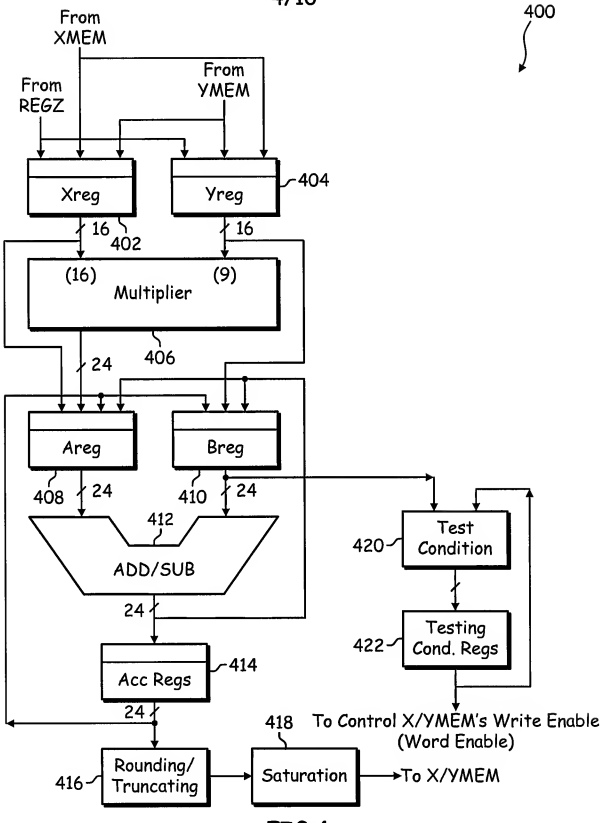
400
From XMEM
From REGZ
From YMEM
Xreg
Yreg
404
16
402
16
(16)
(9)
Multiplier
406
24
Areg
Breg
408
24
410
24
412
ADD/SUB
24
Acc Regs
414
24
Test Condition
420
Testing Cond. Regs
422
418
To Control X/YMEM's Write Enable (Word Enable)
Rounding/ Truncating
416
Saturation
To X/YMEM

FIG.4

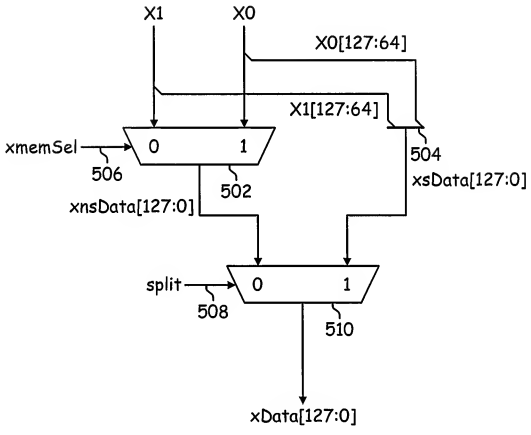
X1
X0
X0[127:64]
X1[127:64]
xmemSel
0
1
506
502
504
xsData[127:0]
xnsData[127:0]
split
0
1
508
510
xData[127:0]

FIG. 5

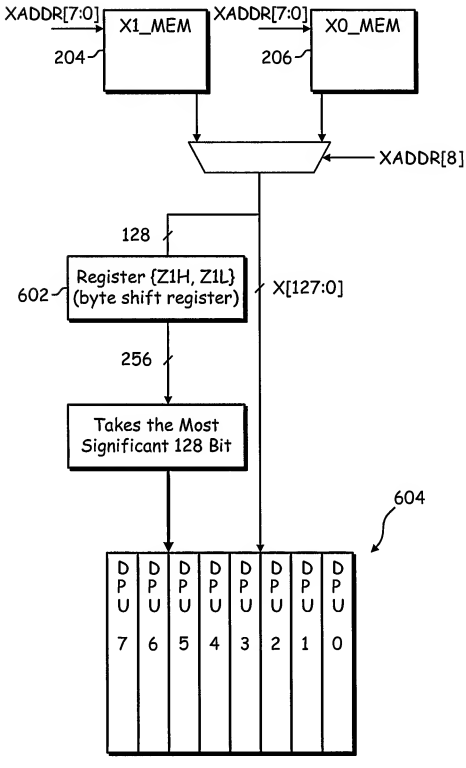
XADDR[7:0]
X1_MEM
204
XADDR[7:0]
X0_MEM
206
XADDR[8]
128
602
Register {Z1H, Z1L}
(byte shift register)
X[127:0]
256
Takes the Most
Significant 128 Bit
604
DPU 7
DPU 6
DPU 5
DPU 4
DPU 3
DPU 2
DPU 1
DPU 0

FIG. 6

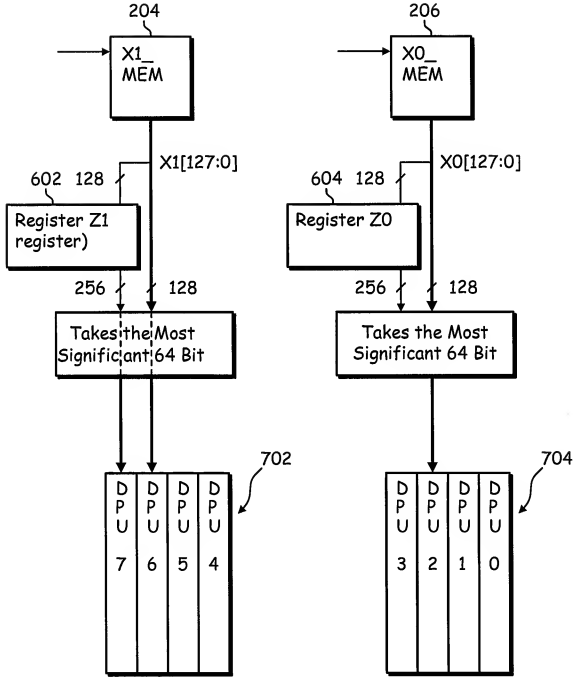
204
X1_
MEM
206
X0_
MEM
X1[127:0]
602 128
Register Z1
register)
X0[127:0]
604 128
Register Z0
256
128
256
128
Takes the Most
Significant 64 Bit
Takes the Most
Significant 64 Bit
702
704
DPU 7
DPU 6
DPU 5
DPU 4
DPU 3
DPU 2
DPU 1
DPU 0

FIG. 7

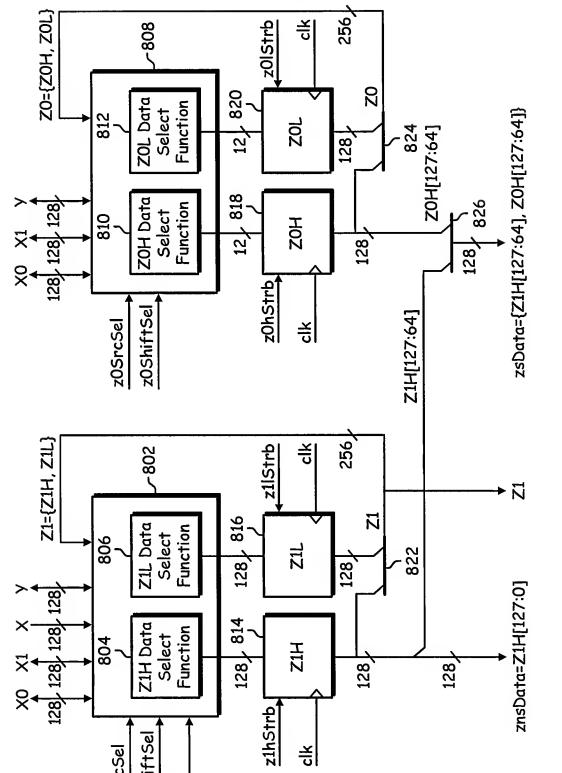
X0
X1
X
Y
128
804
Z1H Data Select Function
806
Z1L Data Select Function
802
Z1={Z1H, Z1L}
z1SrcSel
z1ShiftSel
split
z1hStrb
clk
814
Z1H
816
Z1L
z1lStrb
256
Z1
822
Z1H[127:64]
znsData=Z1H[127:0]
zsData={Z1H[127:64], Z0H[127:64]}
826
824
Z0H[127:64]
z0hStrb
z0lStrb
Z0H
Z0L
Z0
818
820
810
812
808
Z0={Z0H, Z0L}
z0SrcSel
z0ShiftSel

FIG.8

```
1.                LDZS X:(AO)+,     ZH,    NS
2.                LDZS X:(AO)+,     ZL,    NS
3.                LOOP2     #2,     $OUTER_END
4.                MACSZH,   Y:(A1)+,        ACCO, ACC1
5.                LOOP1     #3,     $INNER_END
6.                LDZS RZ,  RZ,     SL
7. $INNER_END     MACSZH,   Y: (A1)+,       ACC1, ACC1
8.                MA   A1,  #-3
9.                LDZS X:(AO)+,     ZH,    NS
10.               LDZS X:(AO)+,     ZL,    NS
11.$OUTER_END     ST   ACC1, Y:(A2)+,       T%1,  SAT,  W%0
12.               EOF
13.               MA   AO,  #-3     <------ This inst always be executed (after EOF)
```

Inner Loop

Outer Loop

FIG. 9

clk 1	clk 2	clk 3	clk 4	clk 5	clk 6	clk 7	clk 8	clk 9	clk 10	clk 11	clk 12	clk 13	clk 14	clk 15
LDZS	LDZS	LDZS	LDZS	LDZS	LDZS									
	LDZS	LDZS	LDZS	LDZS	LDZS									
		Loop2	Loop2	Loop2	Loop2									
			Macs	Macs	Macs	Macs								
				Loop1	Loop1	Loop1	Loop1							
					LDZS	LDZS	LDZS	LDZS						
						Macs	Macs	Macs	Macs	Macs				
							LDZS	LDZS	LDZS	LDZS	LDZS			
								Macs	Macs	Macs	Macs	Macs		
									LDZS	LDZS	LDZS	LDZS	LDZS	
										Macs	Macs	Macs	Macs	Macs

clk 12	clk 13	clk 14	clk 15	clk 16	clk 17	clk 18	clk 19	clk 20	clk 21	clk 22	clk 23	clk 24	clk 25	clk 26
MA	MA	MA	MA	MA										
	MA	LDZS	LDZS	LDZS	LDZS									
		LDZS	LDZS	LDZS	LDZS	LDZS								
			ST	ST	ST	ST	ST							
				Macs	Macs	Macs	Macs	Macs						
					Loop1	Loop1	Loop1	Loop1	Loop1					
						LDZS	LDZS	LDZS	LDZS	LDZS				
							Macs	Macs	Macs	Macs	Macs			
								LDZS	LDZS	LDZS	LDZS	LDZS		
									Macs	Macs	Macs	Macs	Macs	
										LDZS	LDZS	LDZS	LDZS	LDZS

clk 23	clk 24	clk 25	clk 26	clk 27	clk 28	clk 29	clk 30	clk 31	clk 32	clk
Macs	Macs	Macs	Macs	Macs						
	MA	MA	MA	MA	MA					
		LDZS	LDZS	LDZS	LDZS	LDZS				
			LDZS	LDZS	LDZS	LDZS	LDZS			
				ST	ST	ST	ST	ST		
				EOF	EOF	EOF	EOF	EOF	EOF	
					MA	MA	MA	MA	MA	
						INST	INST	INST	INST	INST

INST=(PC=0), if there is no executable function available in the queue;
INST= the 1st inst of next executable function, if there is executable function available in the queue.

note: If the 1st cmd in the queue is "update A0 with 'wait' option", the "update A0" command overwrites the previous MA's result.

FIG. 10